Priscilla Hempel

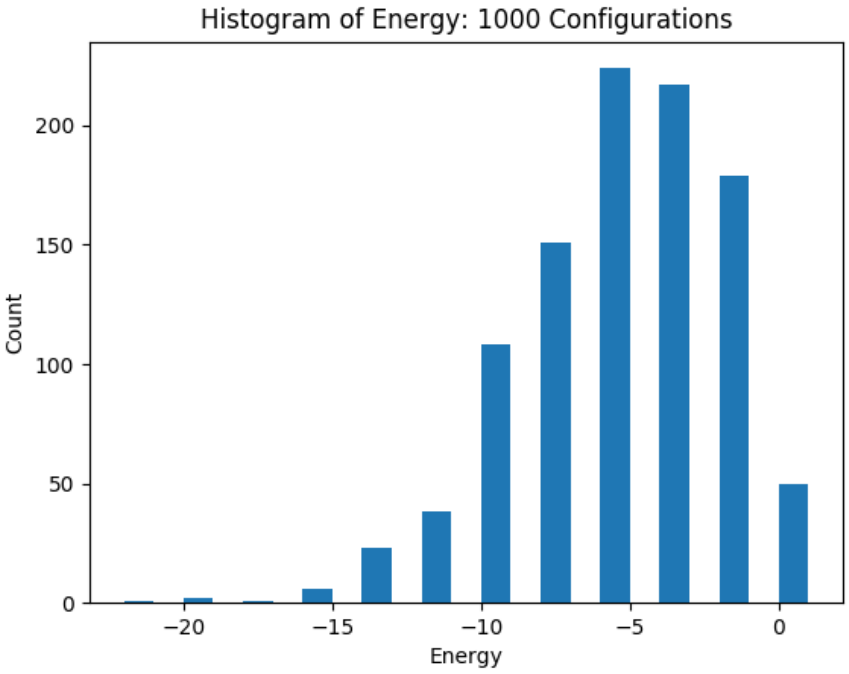
CISC889010: TPCS:ADV ARTIF INTELLIGENCE

PART A

In this section we were asked to read in a file containing a sequence of H and P representing hydrophobic and polar amino acids forming a protein. Additionally, we were asked to build a lattice model to obtain a folded protein configuration and to calculate the energy of the configuration. In my methods, I utilize the **folding**, **calc_energy**, and **get_new_pos** functions I have written to accomplish this.

PART B

In this section, we were asked to implement the metropolis algorithm to simulate a population of configurations that obey the Boltzmann distribution. I accomplished this by utilizing my function **simulate**, which calls my functions **reconfig** and **calc_energy**. Below is the histogram of 1000 iterations, with temperature set to 100. Here we observe that there are only even number energies calculated. This is because in the calc_energy function I double count each topological neighbor's energy.



PART C

In this section, we were asked to optimized our configurations to have the lowest possible energy. Here, we implement the simulated annealing and the metropolis algorithm. First, I initialized a population of random configurations with size = 100. I changed the population size from very small to large and found that larger population sizes yield better optimized configurations and a lower best energy. However, code for larger population sizes take significantly longer to run.

I then chose a random member of this population and created a second generation that will be equal or smaller in size. A random member from this generation is then chosen and a new generation is generated. This process is repeated until our generation size = 1 or 0 OR the temperature has reached 0. Through all this, each legitimate child that is produced in a generation is checked whether it has a lower energy than our current lowest known configuration energy. Children are passed on to a new generation if their energy is deemed appropriate PH_AI.py (lines 376-383). Sometimes, we allow some children with higher energies than their parent so that we may be able to overcome being in a local min of the distribution and find a lower energy. The lowest energy for a configuration that I was able to find through this method is -28. The corresponding lattice configuration can be seen on the next page.

Optimized Configuration (protein index, lattice position)

0,"[0, 0]"
1,"[1, 0]"
2,"[1, 1]"
3,"[0, 1]"
4,"[0, 2]"
5,"[0, 3]"
6,"[1, 3]"
7,"[1, 2]"
8,"[2, 2]"
9,"[3, 2]"
10,"[4, 2]"
11,"[4, 1]"
12,"[4, 0]"
13,"[4, -1]"
14,"[5, -1]"
15,"[5, 0]"
16,"[6, 0]"
17,"[6, 1]"
18,"[6, 2]"
19,"[6, 3]"
20,"[6, 4]"
21,"[6, 5]"
22,"[6, 6]"
23,"[7, 6]"
24,"[7, 5]"
25,"[7, 4]"
26,"[8, 4]"
27,"[8, 5]"
28,"[8, 6]"
29,"[9, 6]"
30,"[9, 7]"
31,"[9, 8]"
32,"[9, 9]"
33,"[10, 9]"
34,"[10, 8]"
35,"[11, 8]"
36,"[11, 9]"
37,"[11, 10]"
38,"[11, 11]"
39,"[11, 12]"
40,"[11, 13]"
41,"[10, 13]"
42,"[10, 12]"
43,"[10, 11]"
44,"[10, 10]"
45,"[9, 10]"
46,"[9, 11]"
47,"[9, 12]"
48,"[9, 13]"
49,"[8, 13]"
50,"[8, 12]"
51,"[8, 11]"
52,"[8, 10]"
53,"[8, 9]"
54,"[8, 8]"
55,"[8, 7]"
56,"[7, 7]"
57,"[7, 8]"
58,"[7, 9]"
59,"[6, 9]"
60,"[6, 10]"
61,"[6, 11]"
62,"[7, 11]"
63,"[7, 12]"
64,"[7, 13]"
65,"[7, 14]"
66,"[8, 14]"
67,"[9, 14]"
68,"[9, 15]"
69,"[10, 15]"
70,"[10, 14]"
71,"[11, 14]"
72,"[11, 15]"
73,"[12, 15]"
74,"[12, 16]"
75,"[12, 17]"
76,"[12, 18]"
77,"[12, 19]"
78,"[12, 20]"
79,"[13, 20]"
80,"[13, 21]"
81,"[13, 22]"
82,"[13, 23]"
83,"[13, 24]"
84,"[13, 25]"
85,"[13, 26]"